Elements of Programming in Java 8

0.5 ECTS

Jaanus Pöial, PhD

Tallinn, Estonia

Estonian Information Technology College

Schedule

Tue, June, 2, 11:20 - 14:20 EDV_A6.08

Wed, June, 3, 12:05 - 15:15 EDV_A2.07

Homework in Moodle

Moodle:

https://cis.technikum-wien.at/moodle24/course/view.php?id=2061

Homepage:

http://fhe.technikum-wien.at/~poeial/

Estonian Information Technology College

Contents

- Quick overview of the Java programming language
- Object oriented programming in Java
- Java 8
  - lambdas, functional interfaces, streams
  - default and static methods in interfaces

Prerequisites: Introduction to programming (basic ideas – program, variable, data type, control flow, subroutine, OOP concepts – class, object, method, inheritance, …)

Java – History

- 1991 - P. Naughton, J. Gosling: project "Green", Sun virtual machine, Oak ==> **Java**
- 1992 - "*7" - video-, cableTV equipment
- 1993 - Mosaic ==> HotJava web browser (P.Naughton, J.Payne)
- 1995 Netscape 2.0 (Java support 1996)
- 1996 - JDK 1.02
- 1997 – JDK 1.1
- 1998 - Java 2 (JDK 1.2..)
- 1999 – J2EE
- 2004 – Java 5, 2006 – Java 6, 2011 – Java 7
- 2014 – Java 8 (25.03.2014)

Features

- C-like syntax, simpler than C++ (more similar to C#)
- No preprocessor
- Object oriented (single inheritance until Java 7, automatic garbage collection, late binding, abstract classes and interfaces)
- Standard and rich APIs (graphics, I/O, data structures, networking, multithreading, …)
- Standard documentation format and documenting tools

Features

- Supported by variety of development environments (IDEs): Eclipse, NetBeans, IntelliJ, …
- Mostly interpretive language, binary representation of a program is platform independent Java bytecode interpreted by Java Virtual Machine (JVM)
- JIT (just-in-time) compilation to machine code is possible
- JVM is used by many other languages (Scala, Clojure, Groovy, ...)

Drawbacks

- Low-level (hardware) programming is hard
- Interpretive => slow (bias)?
- Not flexible enough to create totally new abstractions
- Not suited for beginners, simple programs look complex, drastic learning curve
- Generic programs (programs with type variables) are hard to write (and even hard to understand) in Java 5

General Structure

- Platform API = “technology” (J2SE, J2EE, J2ME, JFX, JavaCard, Java DB, Java TV…)
  Java SE – Java Standard Edition, JDK 8 (JRE included)
- Packages (flat namespace): java.lang, java.util, java.io, javafx.scene, javafx.scene.control …
  Default package is unnamed, package java.lang is always present, other packages need to be imported
- Classes, interfaces, abstract classes – hierarchy (single inheritance for classes, multiple interfaces allowed)
  Class Object is the root of class tree and default parent class if „extends“ clause is missing

Structure of the Class

Data

- Class variables (static), common for all objects, e.g. constants (final)
- Instance variables, individual data, also known as “attributes” or “fields” or “properties”
- Possible inner classes, …

Actions

- Class methods (static), imperative paradigm
  - Constructors to create new objects
- Instance methods (work on objects)

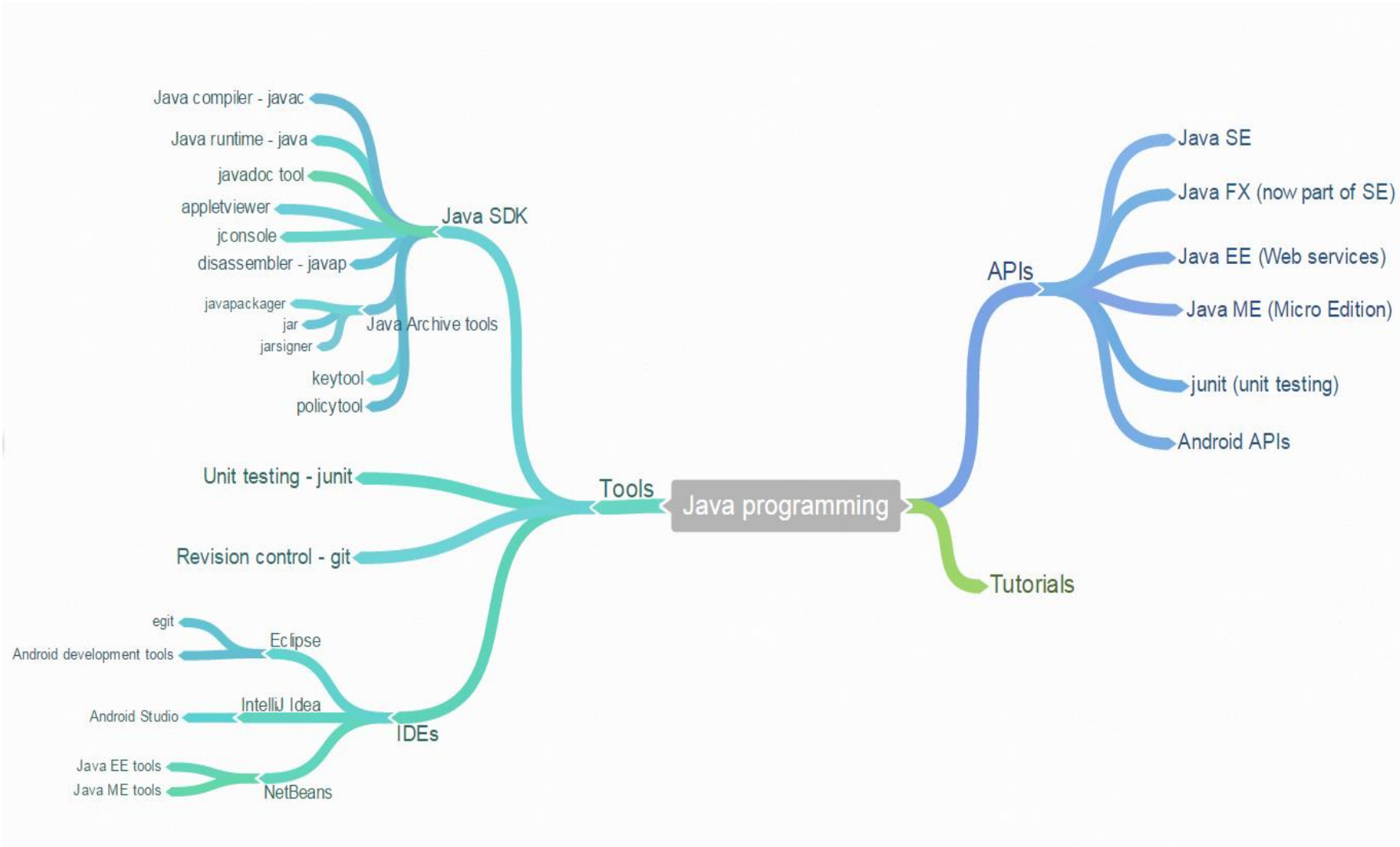
JDK – Java Development Kit

`javac` – Java compiler, X.java => X.class

`java` - Java interpreter, executes X.class

`javadoc` – documentation generator, generates *.html from javadoc comments in program source

…


Java programming
Tools
Java SDK
Java compiler - javac
Java runtime - java
javadoc tool
appletviewer
jconsole
disassembler - javap
Java Archive tools
javapackager
jar
jarsigner
keytool
policytool
Unit testing - junit
Revision control - git
IDEs
Eclipse
egit
Android development tools
IntelliJ Idea
Android Studio
NetBeans
Java EE tools
Java ME tools
APIs
Java SE
Java FX (now part of SE)
Java EE (Web services)
Java ME (Micro Edition)
junit (unit testing)
Android APIs
Tutorials

Examples

Life cycle of a program:

edit, compile, debug syntax, …, run, debug semantics, …,

run, test, …, test …

- `First.java`
- `demo/Example.java`
- `Control.java`

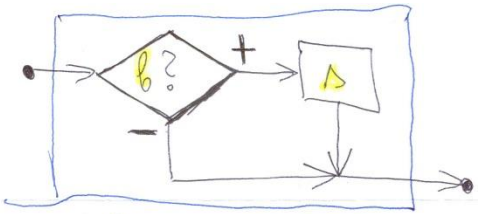
Statements

- block `{ declarations; statements }`
- expression
  - method call: `String.valueOf (56); s.length();`
  - constructor call: `new StringBuilder();`
  - assignment: `variable = expression`
  - complex expression containing operators: `a+b*(c-d)`
- empty statement and labelled statement
- if statement and if-else statement
- switch statement

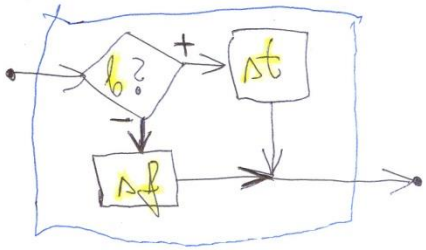
Statements (2)

- for statement
- while statement and do-while statement
- break statement
- continue statement
- return statement
- throw statement
- try-catch construction (try statement)
- `synchronized (object) block;` (synchronized statement)
- assert statement

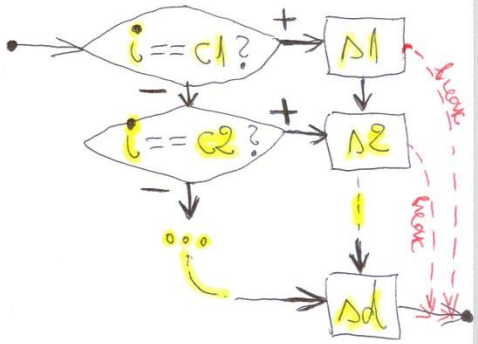
```
if (b) s;
```



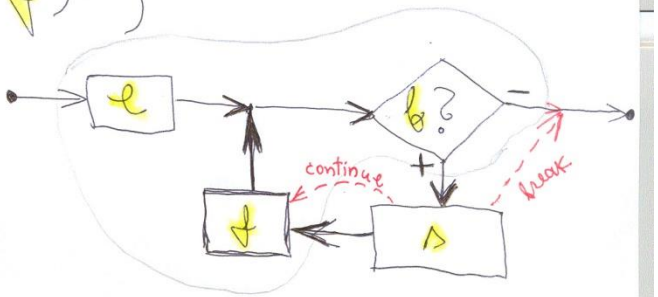

```
if (b) st; else sf;
```



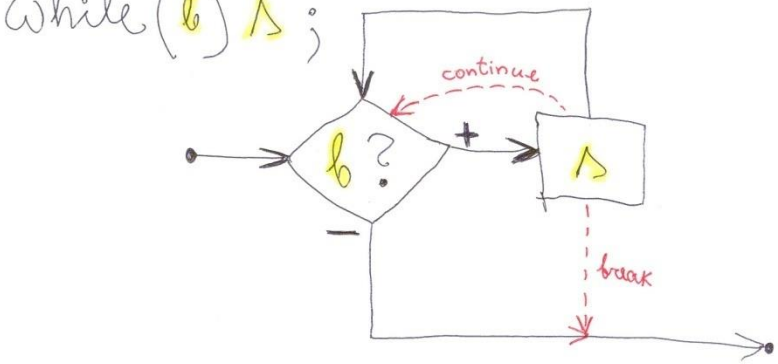

```
switch (i) {
    case c1: s1;
    case c2: s2;
    ...
    default: sd;
}
```

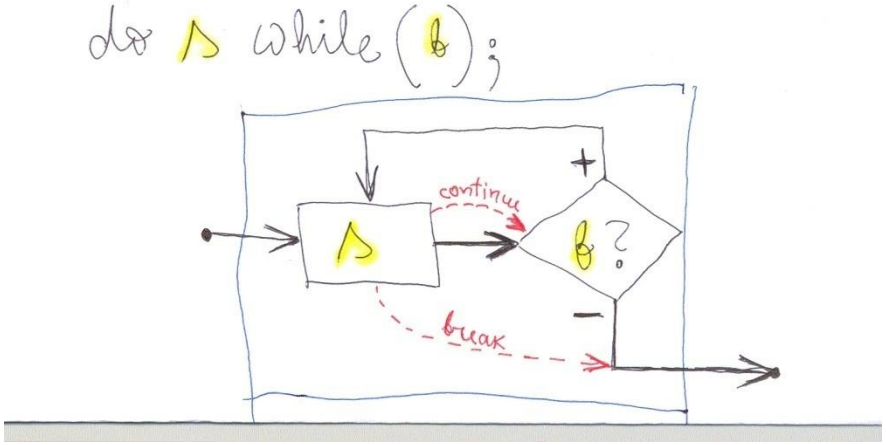



```
for (e; b; f) s;
```



Estonian Information Technology College





Estonian Information Technology College

Examples

- Mswitch
- Mswitchbreak

Data Structures in Java

- Simple variables: primitive types and object types
- Arrays: base type, index, length
- Objects: encapsulate different fields into one instance (like records in “old” imperative languages + methods)
- Collections – built-in tools in Java API to manipulate group of objects: Vector, Hashtable, etc. Java collections framework

Types

- Primitive types: byte, short, int, long, float, double, boolean, char
- Object types:
  - Wrappers: Byte, Short, Integer, Long, Float, Double, Boolean, Character
  - Other API types: String, Object, StringBuffer, …
  - Interface types: Comparable, Runnable, …

Arrays

Array creation and initialization

```
int [] a = { 1, 5, 8 };
```

consists of 3 steps:

```
int [] a;                    // variable declared
a = new int [3];             // memory allocated
a[0]=1; a[1]=5; a[2]=8; // values assigned
int n = a.length;            // array size
```

Array expression:

```
int [] a = new int [] { 1, 5, 8};
```

Multi-dimensional Arrays

- 2-dimensional array is an array of 1-dimensional arrays (NB! these can be of different length):

```
int [][] m;                          // 2-dim array
m = new int [2][];                   // first level
System.out.println (m.length);
m[0] = new int [4];                  // second level
m[0][0] = -8;
m[1] = new int [3];                  // different size
m[1][0] = 9;
```

Objects

Object fields:

```
class Person {
    String surname;
    String firstName;
    Calendar birthDate;
    // etc. whatever we want to record
```

Constructors

```
Person (String sn, String fn, Calendar bd) {
    surname = sn;
    firstName = fn;
    birthDate = bd;
} // constructor

Person() {
    this (“*”, “*”, Calendar.getInstance());
} // default constructor
```

Instance Methods

```
  public String toString() {
    return (firstName + " " + surname
      + " " + String.valueOf (
      birthDate.get (Calendar.YEAR)
      ) + " " + String.valueOf (
      birthDate.get (Calendar.MONTH)
      ) + " " + String.valueOf (
      birthDate.get(Calendar.DAY_OF_MONTH)));
  } // toString


} // Person
```

Usage of Objects

```
public class PersonMain {

    public static void main (String[] args) {

        Calendar bd1 = Calendar.getInstance();

        bd1.set (1959, 04, 30);

        Person p1 = new Person ("Smith",

            "John", bd1);

        System.out.println (p1);

        Person p2 = new Person();

        System.out.println (p2);

    } // main

} // PersonMain
```

Collections

Collection

- *Set* (set, unique elements)
  - HashSet
    - LinkedHashSet
  - *SortedSet* (ordered set, unique elements)
    - TreeSet
- *List* (dynamic, indexed, multiple copies allowed)
  - ArrayList
  - LinkedList
  - Vector (legacy API, similar to ArrayList)
- *Queue* (since Java 5, not discussed here)

Collections

Map ("key-value" pairs)

- HashMap
  - LinkedHashMap
- *SortedMap*
  - TreeMap
- Hashtable (legacy API)
- WeakHashMap (allow garbage collection)

Iterator (to find the next element)

Enumeration (legacy API, similar to Iterator)

Iterable (has iterator)

- *Collection*

Collections

Comparable

which of two elements is "bigger"

```
public int compareTo (Object o)
```

o1.compareTo (o2) =(/ -1, if o1 < o2
0, if o1 == o2
\ 1, if o1 > o2

Arrays

static utilities – asList, search, sort, fill, ...

Collections

static utilities – search, sort, copy, fill, replace, min, max, reverse, shuffle, ...

Java Command Line

- Javac – compiler

```
javac Cunit.java
javac -cp classpath Cunit.java
javac my/package/Myclass.java
```

- Java – interpreter

```
java Cunit any text you like to pass
java -cp classpath Cunit
java my/package/Myclass
java my.package.Myclass
```

Junit – www.junit.org

```
javac -cp .:junit-4.12.jar ClassTest.java

java -cp .:junit-4.12.jar org.junit.runner.JUnitCore ClassTest
```

Examples

Eclipse

- www.eclipse.org

  Eclipse for Java Developers (EE not needed)

  - Make a Java project
  - Add a class to the project
  - Run the program